Classification of Mars Terrain Using Multiple Data Sources

Alan Kraut[1], David Wettergreen[1]

ABSTRACT. Data about Mars are being collected faster than they can be analyzed by geologists for the creation of geologic maps. Because of this, automatic analysis of data would be very useful. We develop a method for using data from multiple sources to classify areas of Mars terrain. Each scene is over segmented into superpixels, and a feature vector is developed for each superpixel. Several classification algorithms are examined for assigning a class to each superpixel. These algorithms are trained using three different manual classifications with between 2 and 6 classes. Automatic classification accuracies of 50%-80% are achieved in leave-one-out cross-validation across 20 scenes using a generalized boosting classifier.

1. INTRODUCTION

The creation of geologic maps is critically important for planetary science. Geologic maps identify spatial trends that indicate formative processes. They can represent mineralogical properties, history of the area, structure of the terrain, or many other things [1]. These maps are a way of distilling information about an area of terrain to be easily referenced. Planetary scientists painstakingly infer separate units of surface material from all available information—which may include orbital images, soil samples, and elevation maps—to create them. We have created a tool for using orbital images of Mars to automatically create first-pass approximations of geologic maps.

1.1. Motivation.

Over the past decade huge amounts of data have been collected about Mars in the form of orbital images. In some areas, these images have been analyzed by planetary scientists to create geologic maps, but the sheer amount of data means that most of Mars remains unmapped, and much of the data is nearly untouched.

The United States Geological Survey (USGS) is currently the primary organization creating geologic maps of Mars. They have released maps of fewer than 30 regions of Mars [2]. While some of these are quite extensive, they leave most of the surface of Mars unmapped.

Because there is so much data available about Mars, a system that could draw attention to specific regions of potential interest would be extremely valuable. We believe the best way to accomplish this is to create automatic geologic classifications based on training examples. This is, given a set of training scenes classified in any way, we will classify a new scene using the same set of classes. Thus a scientist could find areas of Mars with a high density of a desired terrain type by hand classifying some training examples, training the system on those examples, and letting it classify the rest of the surface of Mars. Areas that have a high density of that terrain type could then be further examined by the scientist.

[1] Robotics Institute, Carnegie Mellon

1.2. Prior work. Tomasz Stepinski has done work in automated recognition of Mars landforms based on elevation data [3]. The work was quite successful in determining the kind of land formation, such as craters and ridges. However, because it only uses elevation data it would be insufficient for creating geologic maps. Geologic maps can be based on the physical structure of the terrain, but can also be based on factors such as mineral content, and how the soil was deposited. A classifier that uses only elevation data would be incapable of examining features such as absorption spectra, which are a key indicator of mineral content.

1.3. Problem and approach. Our goal is to automate the process of creating a geologic map of the surface of Mars. This should use an arbitrary number of orbital images of the same area to create an automatic classification. This is not expected to be as accurate as a hand classification, but it should provide a useful idea of the character of different areas.

We pose the task of map making as finding a solution to the segmentation problem of dividing the scene into a large number of areas, and the classification problem of assigning a class to each of these segments. The structure of our method is shown in Figure 1.

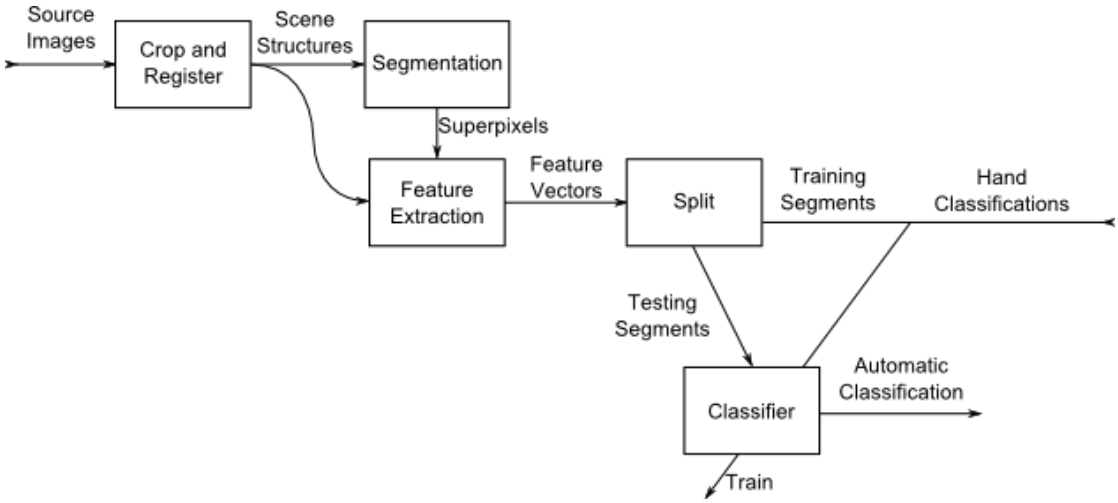


Figure 1: Flow chart showing structure of classification method.

To accomplish the segmentation task we use a superpixel segmentation. A superpixel segmentation has high recall, so that nearly all actual edges between map units are represented, but can have low precision, meaning that having edges in the segmentation that are not actually edges between map units is okay. Our method for creating these segmentations is discussed in Section 2.

We then perform the classification task assuming that each superpixel will be composed of a single class. The automatic classification is performed on a feature vector for each superpixel. This consists of statistics about the image information within the superpixel, and is detailed in Section 3. We examine Bayesian and boosting classifiers for performing

this step. Additionally, belief propagation is examined as a way to use the spatial relations between superpixels. The classification algorithms are discussed in Section 5.

2. SUPERPIXEL SEGMENTATION

Fundamentally each pixel in the test image needs to be assigned a class. However, trying to classify a single pixel is often infeasible [4]. We first over-segment the image into chunks that were each assumed to be of one uniform class, called superpixels. This has been shown to be useful in allowing higher level reasoning in classification. For example, Greg Mori showed the ability of superpixel segmentations to assist in matching sections of images to models, by examining many possible combinations of superpixels [5].

2.1. Initial segmentation.

Our code is based on a two-step segmentation algorithm developed by Mori. In this algorithm a random sampling of pixels in the image is taken, and a graph is created with each of those pixels as a node. The edge weight between two pixels is set to be the negative exponential of the maximum boundary probability (P^b) between them in the image. The P^b is calculated as a sum of exponentials of the local intensity and texture gradient of the image. This graph is made sparse by removing edges with a weight below a certain threshold. A standard normalized cut (n-cut) algorithm is then performed on this graph [6]. The results are interpolated to the remaining pixels.

2.2. Recursive segmentation.

Because of the computational complexity of n-cutting, it is frequently impractical to create the desired number of superpixels using a single pass. To overcome this limitation, a small number of superpixels are created using the method described above, and subsequently divided into smaller superpixels. The algorithm is called recursively on each superpixel created at a single step, with a desired total number of superpixels based on the area of the current region. This is done until enough superpixels have been created. The results of this algorithm are shown in Figure 2.

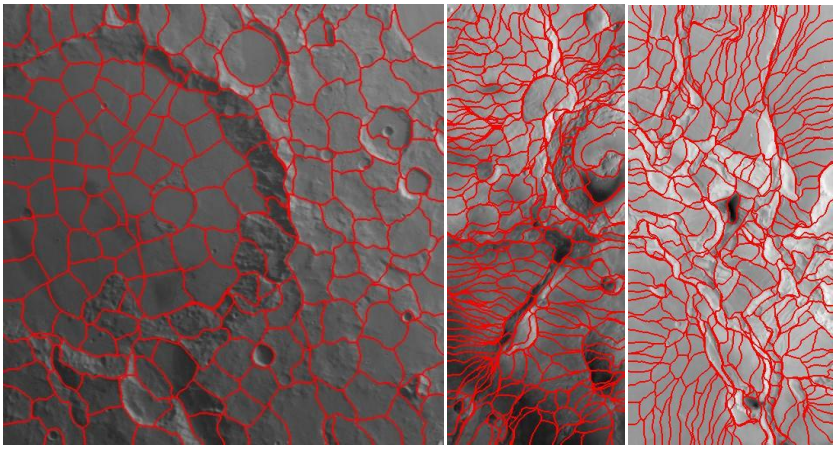


Figure 2: Examples of segmentation algorithm on three scenes.

3. FEATURE GENERATION

In order to classify an image, we first create a vector of numbers to describe the section to be classified. This is a feature vector. We create a feature vector for each superpixel. One possible feature vector would simply be a vector of the pixel values from all channels in the superpixel. However, in order to create an effective classifier, it is necessary to create individual features that are correlated with the class of the superpixel.

In this section we will describe how we take a multi-channel image and create a feature vector for a single superpixel. Each of these feature extractors can be applied to an arbitrary channel (or sometimes multiple channels). When applied to different channels they represent different sorts of information about the terrain. Table 1 summarizes the features we developed, and they are described in greater detail below.

Feature	Number of channels used	Number of elements
Mean	1	1
Standard Deviation	1	1
Mean of Ratios	2	1
Laplacian	1	1 per scale
Laplacian of Ratios	2	1 per scale
Texton Histogram	1	16
MR8 Filter Bank	1	8

Table 1: Feature extractors developed.

3.1. Mean and standard deviation. The simplest features we use in our feature vector are the robust mean and standard deviation of a given channel across the super pixel. The robust mean is calculated by removing the highest and lowest 10% of the data, and finding the mean on the remaining data.

3.2. Ratio of channels. Another feature used is the ratio of two channels. Specifically, the log of the ratio of two channels is taken pixel by pixel across the image. A logarithm is used so that the scale of features in areas where the numerator image is more intense than the denominator image will be the same as the scale in areas where the reverse is true. This allows us calculate one ratio feature per pair of channels, instead of two. After these values are computed across the image, the mean is taken over the superpixel.

This feature captures information about the relation between two channels, and is expected to be most relevant when calculated using a pair of channels of two different wavelengths. In order to capture geologic information it is important to consider the relation between different wavelengths. Geologists use absorption spectra and emission spectra to identify different compounds. Unfortunately we do not have enough data to actually calculate spectra, but by calculating the ratio of responses at different wavelengths, we expect to capture information relevant to color and mineral composition. In the event that particular known minerals are being looked for, features could be

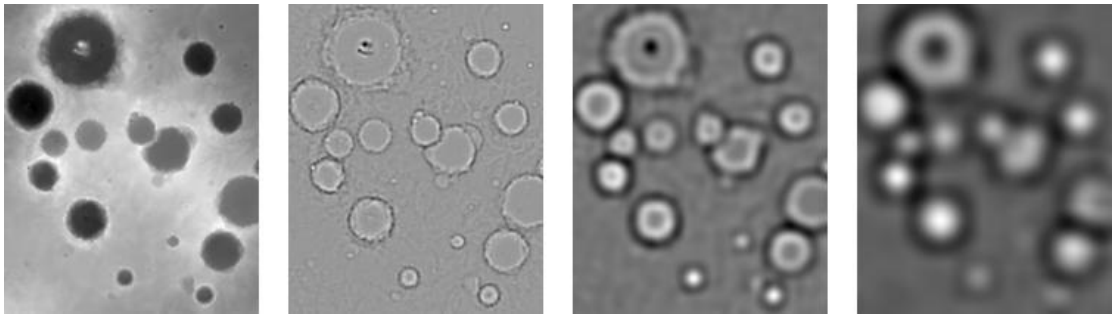
developed that correlate with how well the ratios of two or more wavelengths match a specific emission spectrum.

3.3. Laplacian at multiple scales. The Laplacian is a filter which measures the difference of one area of an image from the surrounding area. We use a difference of Gaussians approximation of the Laplacian filter [7], with a square filter with 2σ elements on either side of the center. For example, Table 2 shows the filter used for $\sigma = 0.5$.

0.4038	0.8021	0.4038
0.8021	-4.8233	0.8021
0.4038	0.8021	0.4038

Table 2: 3x3 Laplacian filter approximation

This filter only captures variations of a particular scale. Specifically, any given filter will have a strong response for features with a radius of about σ. In order to capture variations in the image at a variety of scales, we use $\sigma = 0.5(2^n)$, with n between 2 and 8. Examples of an image with the Laplacian filter run on it at various scales are shown in Figure 3.



(a) (b) (c) (d)

Figure 3: Elevation map (a), and Laplacians computed with n=1 (b), n=3 (c), and n=4 (d).

Once the response of the Laplacian filter is computed, its mean is taken over the superpixel to reduce it to a single-value descriptor. This is done once at each scale specified, so that each scale has a single feature in the feature vector. It is additionally run on the image created by taking the log ratio of channels, as described in Section 3.3.

This feature captures variation in the image. This is useful because it makes it robust to lighting changes, gross changes in elevation between different sections of Mars, etc. For luminosity channels this will represent the apparent lightness or darkness as compared to the surrounding terrain, and for the elevation channel it will capture local depressions or elevations in the terrain.

3.4. Texture features. One thing that is useful when classifying images is a representation of the texture content of an image. Texture is a perceptually complex feature, and is usually represented in image analysis by textons, which are archetypal responses to a set

of filters. Textons, particularly histograms of texton frequencies, have been shown to be effective at discerning between different textures [8]. This is especially true when the textons are generated from examples of textures to be classified.

3.4.1. Filter bank. We use the MR8 filter bank [8] (Figure 4) for the creation of textons. The MR8 filter bank consists of bar filters at six different orientations and three scales, with both edge and symmetric filters, as well as a Gaussian and a Laplacian filter.

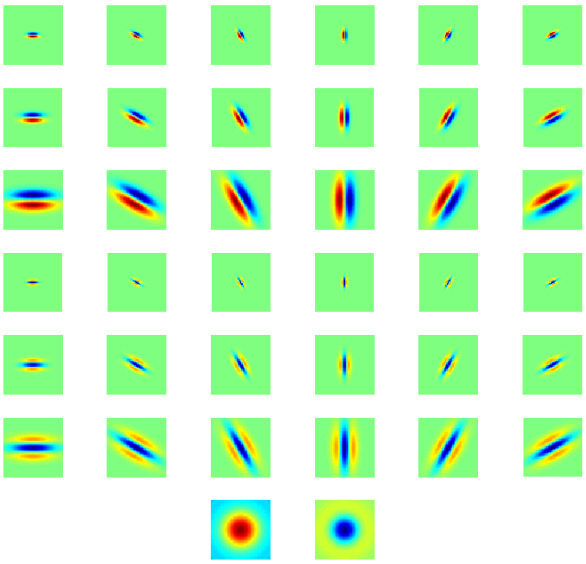


Figure 4: MR8 filter bank. Feature vector is maximum response to each row of 6 filters, plus the response to each of the bottom two filters.

We use the MR8 filter bank for two reasons. First, taking the maximum response across orientations allows it to be rotation invariant. This is desirable because terrain type should not depend on its orientation. Second, it captures different scales of texture. We would like our filters to be able to differentiate between different scales because two geologically distinct regions may have similar textures, but at different scales.

3.4.2. Texton generation. Textons are weighted sums of the filters in the filter bank, and represent archetypes of local image response. They are calculated by using k-means clustering on the 8-dimensional points generated by taking the MR8 response centered at a given point in an image. Filter responses are taken across a large set of images, and the k cluster centers of response vectors from all the images are used as the textons. We used approximately 500 HRSC images of Mars in the visible and near IR wavelengths as the set of images from which to compute textons.

3.4.3. Texture feature generation. Two types of texture features are generated for each superpixel. First, the average MR8 filter response across the superpixel is used for 8

features. Second, a histogram of texton frequencies across the superpixel creates another feature for each texton being used.

In order to create the texton frequency histogram, each pixel in the image is first assigned a texton. This is done by calculating the filter response at that pixel, and assigning the texton with the nearest filter response by the L2 distance. Once each pixel has an assigned texton, a histogram is created by counting the number of pixels associated with each texton inside the superpixel. This histogram is normalized to make it invariant to superpixel size, and each bin is used as a feature for the superpixel.

This set of features represents how often different textons appear in the superpixel. This can help distinguish between different types of terrain, such as steep cliffs, sand dunes, and cracked land.

3.4. Features used. Table 3 summarizes the elements of our final feature vector. Most channels are cropped and registered images taken from Mars orbital assets. The exception to this is "MOLA Slope", which is the magnitude of the gradient of the MOLA height map.

Feature	Channels and Parameters	# elements
Mean	HRSC IR, Red, Green, Blue, and ND, MOLA Slope	6
Standard Deviation	HRSC ND, MOLA Slope	2
Mean of Ratios	All pairs of narrow-band HRSC channels	6
Laplacian	MOLA Elevation, n=[2,4,5,6,7]	5
Laplacian	HRSC ND, n=[2,4,6,7,8]	5
Laplacian of Ratios	HRSC Blue, HRSC IR, n=[2,4,5,6,7]	5
Texton Histogram	HRSC ND	16
MR8 Filter Bank	HRSC ND	8

Table 3: Elements of feature vector.

This is the feature vector used for all classification methods we examined. We include all the features we believe to be salient, but attempt to exclude redundant features. The desire to exclude redundant features comes from concerns about runtime and overfitting. Especially in training a boosting classifier (see Section 5), adding more features increases the run time significantly. Also, our set of training images was not as large as would be desirable, so adding spurious features would create a risk of overfitting. The more uninformative features are added, the more likely it becomes that a correlation will be observed in the training sample that is not representative of the overall population. Large training sets help mitigate this problem by increasing how representative the training sample is of the population. If the method is being used in such a way that it is trained once and then used to classify a large number of images, it would likely be desirable to include a larger set of features, because the cost is not as great.

4. Manually Labeled Data

Automatic classification methods rely on training data. While there are geologic maps of various regions of Mars, they do not cover all the regions we were testing. In order to create training data, we hand-labeled our sample scenes in a variety of ways. We created one hand classification based on coarse terrain features, one based on regions exhibiting Aeolian deflation, and one based on coarse classes distilled from existing USGS maps.

4.1. Terrain features. One hand classification we created has four classes, corresponding roughly with lowlands (such as crater basins, and valley floors), slopes (such as the edges of craters and valleys, and steep ridges), plains/plateaus, and volcanos (large, smooth mountains). We expect that these classifications would be strongly correlated with features derived from elevation maps. Examples of this hand classification can be seen in Figure 6.

Figure 5: Terrain legend

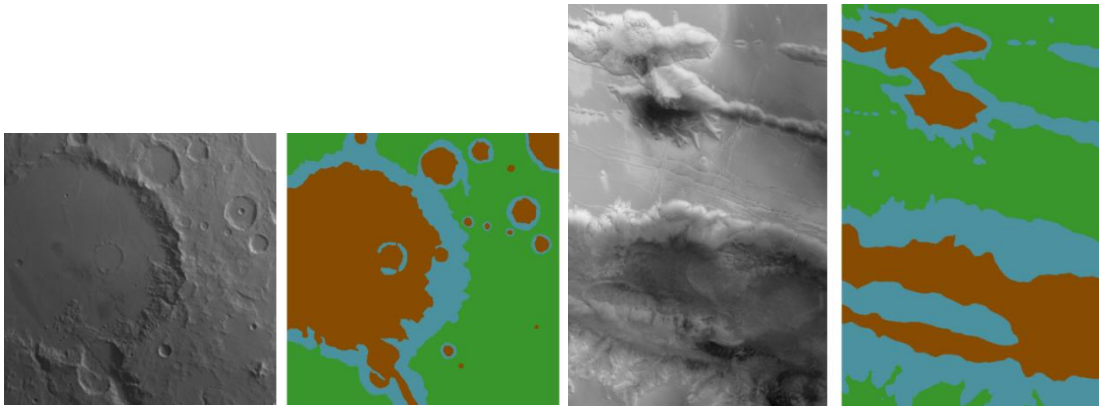


Figure 6: Two sample images with terrain feature classifications.

4.2. Aeolian deflation. The removal of very fine dust and sand by wind processes is known as Aeolian deflation. On Mars these regions can be seen as areas with a distinct deep purple color. Areas of Aeolian deflation were fairly sparse. This classification is expected to utilize features such as ratios of wavelengths that are correlated with color. It is also a test for the case of an uncommon class.

Figure 7: Aeolian legend

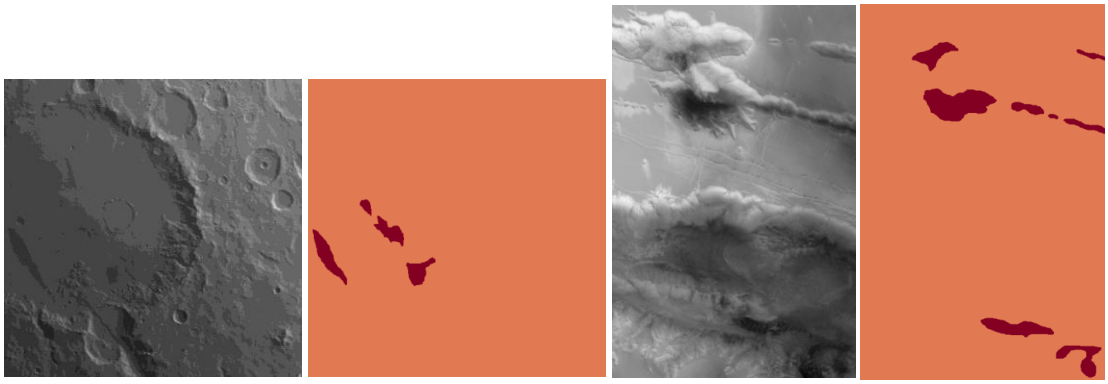


Figure 8: Two sample images with Aeolian deflation classifications.

4.3. Geologic classification. The last hand classification we created is based on the broad categories we saw repeatedly in USGS maps of Mars. Where possible this classification was taken directly from those maps, but we extrapolated to the best of our ability for other regions. This classification consists of six classes, corresponding to vallis materials, crater materials, other steep slopes (corresponding with class HNw on USGS Mars geologic maps), crater fill, plains, and mountainous materials. This is expected to be the manual classification that is closest to the expected use case of this method. It is a complex classification in that no single kind of feature is likely to be able to do a good job distinguishing between all five classes. Some examples of this classification can be seen in Figure 10.

Figure 9: Geologic legend

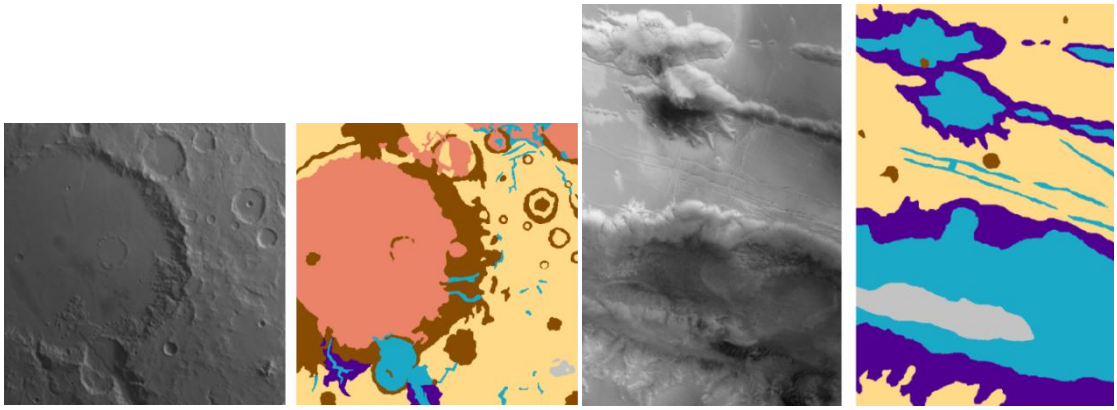


Figure 10: Two sample images with USGS-based classifications. The image on the left (Gusev Crater) has an existing USGS map.

5. Classification

We classify the data using a naïve Bayes classifier and a boosting classifier, and use the results from both classifiers to inform a belief propagation network.

5.1. Naïve Bayes classifier. A naïve Bayes classifier operates by calculating the probability of an example being of a given class using Bayes' Rule, and selecting the class with the highest probability. Specifically, we want to compute $P(C|F)$, where C is the segment's class, and F is its feature vector. For a single class c_i and feature f_k we can write this as

$$P(c_i|f_k) = \frac{P(f_k|c_i)P(c_i)}{P(f_k)}.$$

When there are many of these features, we compute the probability of the joint as the product of the probabilities, i.e.

$$P(c_i|F) = P(c_i)\prod_k \frac{P(f_k|c_i)}{P(f_k)}$$

Once the probability of each class has been computed, the class with the highest probability can be selected. Because all the $P(f_k)$ will be the same for each class, this can be written as

$$x = \underset{i}{\operatorname{argmax}}\left(P(c_i)\prod_k P(f_k|c_i)\right)$$

Where x is the assigned class. $P(c_i)$ is calculated from the training data as the fraction of the examples that are of class i. $P(f_k|c_i)$ is calculated by adaptively assigning a set of ranges to each feature, and then finding the fraction of examples of class i that has the feature in each range.

5.2. Boosting classifier. Boosting is a method for creating a single classifier by combining many weak classifiers—classifiers which are more accurate than chance, but would not be good enough for the desired application by themselves. One very common boosting algorithm is AdaBoost. AdaBoost was developed by Freund and Schapire [9], and has since become a well studied algorithm which is commonly used. It trains many simple classifiers (called base classifiers), assigns them weights based on their accuracy in classifying the training set, and uses a weighted majority vote. This method has proven highly successful in a wide variety of domains when a binary decision needs to be made.

However, AdaBoost is specific to a binary classification task. We would like to be able to classify our data into an arbitrary number of classes. That is, there should be no hard upper limit on the number of classes present in a hand segmentation used to train the classification method. Because of this, we use an extension of AdaBoost. We modeled this extension off the GrPloss algorithm [10].

The GrPloss algorithm creates an n-class classifier for data, given labeled training data. It provides an initial weight vector to the training examples. At each iteration it trains a base classifier. This classifier takes a feature vector, and returns an n-element vector of class probabilities. The classifier is then assigned a weight based on its weighted error (the classification error on the training set using the current weight vector), and examples which were misclassified have their weight increased.

To classify a new example, a weighted average of the class probabilities from each of the base classifiers is taken, and the maximum probability class is chosen.

We use a base classifier known as a decision tree. A decision tree makes a series of up to N binary decisions, resulting in 2^N possible outcomes, where N is the depth of the tree. We use trees of depth 2. Deeper trees would result in a more discriminating base classifier, at the cost of exponentially higher training time, and a greater risk of overfitting.

Each node of the decision tree divides the data it receives into two categories based on a threshold on a single feature. If that feature is above the threshold, it is sent along one branch of the tree to the next node, and if the feature is below the threshold, it is sent along the other branch. Once a leaf is reached the decision tree returns a pre-set probability vector for that leaf. The probability vector is set based on the training examples which reach that leaf.

5.3. Belief propagation. In order to incorporate spatial information we use a loopy belief propagation framework. Belief propagation (BP) is an algorithm that finds a solution for the most likely class of each node in a directed graphical network [11]. It is often used to find boundaries and smooth results when a good initial guess for node classification can be given. For BP, each node has an associated vector of label evidence, $\phi_i(x_i)$, the probability of node *i* being of class x_i, and each edge has a compatibility matrix, $\psi_{i,j}(x_i, x_j)$, the probability of node *j* being of class x_j, given that node *i* is of class x_i.

At each time step a message is sent on each edge leading away from a node based on all messages leading to that node from any other direction. Specifically, the message from node *i* to node *j*, $m_{i,j}(a)$, is given by:

$$m_{i,j}(x_j) = \sum_{x_i} \phi_i(x_i)\psi_{i,j}(x_i, x_j) \prod_{k \epsilon N(i) \setminus j} m_{k,i}(x_i).$$

Once these messages converge, the belief at each node is calculated as

$$b_i(x_i) = \phi_i(x_i) \prod_{k \epsilon N(i)} m_{k,i}(x_i)$$

and the class with the largest value at node *i* is selected.

When being applied to results from the Bayesian classifier, the ψ matrix is calculated using the edge probability (P^b) already calculated in determining the superpixels, normalized to be in the range [0,1]. It is given by

$$\psi_{i,j}(x_i, x_j) = \begin{cases} (1 - P^b_{i,j})\theta_{x_i,x_j} & x_i = x_j \\ P^b_{i,j}\theta_{x_i,x_j} & x_i \neq x_j \end{cases}$$

where θ_{x_i,x_j} is the probability of x_i and x_j bordering each other, and $P^b_{i,j}$ is calculated as the average of P^b across all pixels on the border between superpixels *i* and *j*.

When applied to the results from the boosting classifier, the compatibility matrix is formed such that the initial classification boundaries would not change. Specifically, it is given by

$$\psi_{i,j}(x_i, x_j) = \begin{cases} (1 - D_{i,j})\theta_{x_i,x_j} & x_i = x_j \\ D_{i,j}\theta_{x_i,x_j} & x_i \neq x_j \end{cases}$$

where $D_{i,j}$ is 1 if the two superpixels are the classified differently in the initial classification, and 0 if they are the same. This is done because the boosting classifier tends to create boundaries between classes in close to the right place, but sometimes sets the entire region as the wrong class.

5. Results and Conclusion

We tested the classification method using leave-one-out cross validation on scenes. We used 20 scenes, each divided into roughly 300 superpixels, so each trial trained on approximately 5700 superpixels, and classified approximately 300 superpixels. The results are summarized in Table 4.

	Bayes Classifier		Boosting Classifier	
	Without BP	With BP	Without BP	With BP
Terrain	54%	60%	66%	64%
Aeolian	88%	90%	88%	87%
Geologic	47%	51%	49%	49%

Table 4: Average classification accuracies over all test images

Overall the boosting classifier performed slightly better than the Bayes classifier. The naïve Bayes classifier was marginally improved by the inclusion of belief propagation, while the boosting classifier was not improved by belief propagation. We believe this to be because turning the output of the naïve Bayes classifier into a probability vector for use in belief propagation is straightforward and principled, due to the probabilistic nature of Bayes classifiers. On the other hand, the boosting classifier returns scores, not probabilities, and it is not clear how these should be translated to the evidence vector.

Some typical results from the boosting classifier are shown below. Figure 11 shows a scene using the terrain feature classification. This shows that some of our features successfully incorporate spatial information. Specifically, the strip of correctly classified lowlands around the edge of Gusev crater is roughly the width of the largest Laplacian of elevation used in the classification. This indicates that the classifier could be improved by the inclusion of features that use data from a larger spatial expanse of the image.

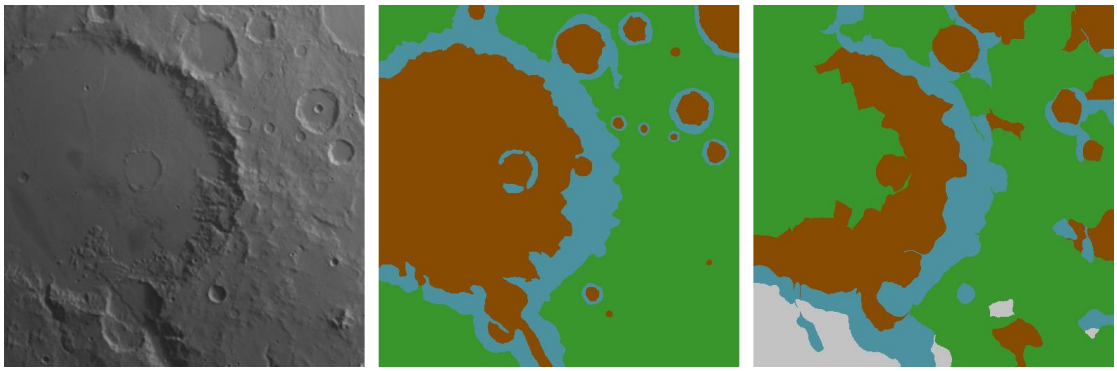


Figure 11: The terrain features hand classification (center) and automatic classification (right) for the Gusev Crater scene.

Figure 12 shows a typical result from the geologic classification. This demonstrates substantial confusion between the crater material and slopes classes. . Currently it is likely that the examples for the valley slopes and crater material classes form overlapping balls in the 52-dimensional space of the feature vector. It is possible that there is another feature that could be computed in which these balls have a large separation compared to their width. If such a feature exists, adding it would make the classes much more distinguishable. It is also possible that these two classes should be divided differently in the hand classification, such that each new class is more internally consistent, and has less overlap with the others. For example, creating a subclass of crater material for ejecta blankets could improve the classification accuracy.

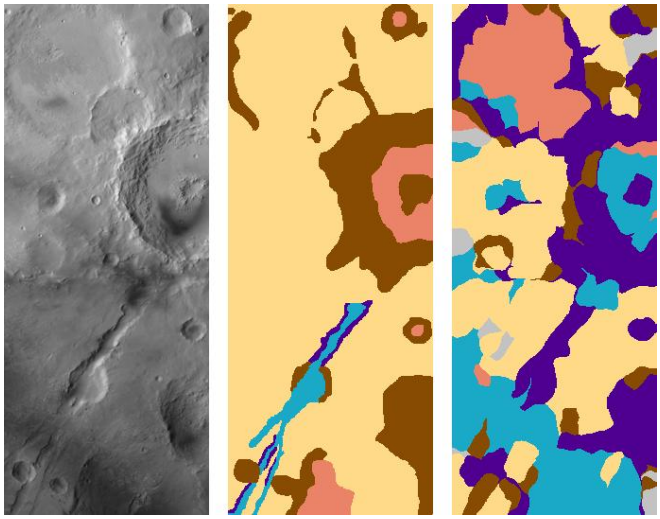


Figure 12: The classification based on USGS maps for the Nili Fossae region.

Overall we believe that this method has much room for improvement, but that it demonstrates the ability to effectively combine different types of information and to generalize to different kinds of classification. There are two key areas for future work on this project. It needs to be able to incorporate information about how different parts of the scenes are arranged spatially. Ideally there would be some way to represent and learn information such as which classes are likely to be near each other, and under what circumstances. The alrogithm also needs to be able to use more data sources. Currently the method does not allow for partial information data sources. That is, in order for a data source to be used for any superpixel, it needs to have valid information for all superpixels. This effectively limits us to using a single instrument which does not have complete coverage of Mars. Solving this problem would allow much more data to be used in making classification decisions.

REFERENCES

[1] B. Brodarik and J. Hastings. An Object Model for Geologic Map Information. Advances in Spatial Data Handling, 2002.

[2] http://geopubs.wr.usgs.gov/docs/wrgis/mars.html

[3] T. Stepinski. Machine Learning Tools for Automatic Mapping of Martian Landforms. IEEE Intelligent Systems, 2007.

[4] X. He, R. Zemel and D. Ray. Learning and Incorporating Top-Down Cues in Image Segmentation. Computer Vision ECCV, 2006.

[5] G. Mori. Guiding model search using segmentation. IEEE International Conference on Computer Vision, 2005.

[6] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(8), August 2000.

[7] S. Gunn. On the discrete representation of the Laplacian of Gaussian. Pattern Recognition, 32, 1999.

[8] M. Salahuddin, M. Drew and Z. Li. A Fast Method for Classifying Surface Textures. IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.
[9] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, No. 5, 1997.
[10] G. Eibl and K-P. Pfeiffer. Multiclass boosting for weak classifiers. Journal of Machine Learning Research, 6:189–210, 2005.
[11] K. Murphey, Y. Weiss and M. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study. Uncertainty in AI, 1999.